PROJECT DEVELOPMENT PHASE
SPRINT2

Date	24 October 2022
Team ID	PNT2022TMID17788
Project Name	Smart Lender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	20 Marks

MODEL BUILDING

ibm.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
[ ] #Importing The Libraries
```

```
[ ] import pandas as pd
    import numpy as np
    import pickle
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    import sklearn
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import RandomizedSearchCV
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score,ConfusionMatrixDisplay,precision_score, recall_score
    from sklearn.linear_model import LogisticRegression
```

```
#Reading The Dataset
```

```
[ ] data = pd.read_csv('C:\\Users\\rajaa\\naalaiya thiran\\dataset\\train_u6lujuX_CVtuZ9i.csv')
    data
```

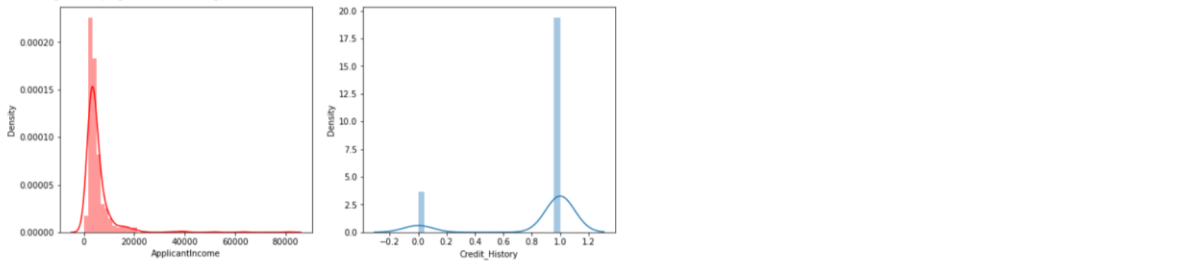
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 13 columns

```
[ ] #Uni-Variate Analysis
```

```
[ ] plt.figure(figsize=(12,5))
    plt.subplot(121)
    sns.distplot(data['ApplicantIncome'], color='r')
    plt.subplot(122)
    sns.distplot(data['Credit_History'])
    plt.show()
```

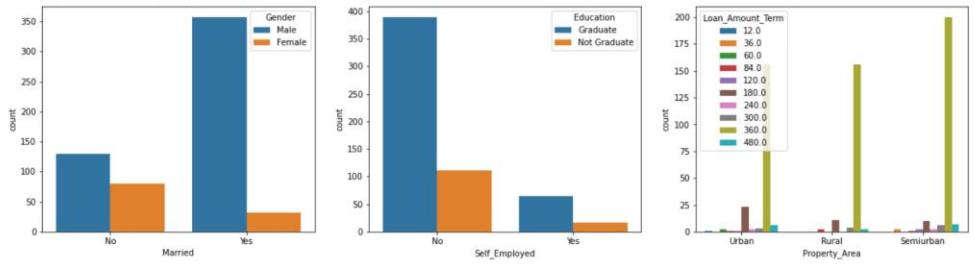
```
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your c
  warnings.warn(msg, FutureWarning)
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your c
  warnings.warn(msg, FutureWarning)
```




```
[ ] #Bivariate Analysis
```

```
[ ] plt.figure(figsize=(20,5))
    plt.subplot(131)
    sns.countplot(data['Married'], hue=data['Gender'])
    plt.subplot(132)
    sns.countplot(data['Self_Employed'], hue=data['Education'])
    plt.subplot(133)
    sns.countplot(data['Property_Area'], hue=data['Loan_Amount_Term'])
```

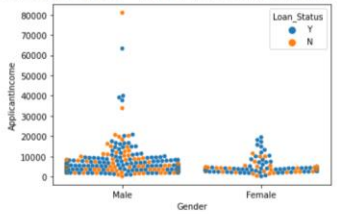
```
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argume
  warnings.warn(
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argume
  warnings.warn(
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argume
  warnings.warn(
<AxesSubplot:xlabel='Property_Area', ylabel='count'>
```



[ ] #Multivariate Analysis

[ ] sns.swarmplot(data['Gender'], data['ApplicantIncome'], hue = data['Loan_Status'])

```
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argu
  warnings.warn(
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 67.5% of the points cannot be placed; you may want to decrease the size of the markers or use stripp
  warnings.warn(msg, UserWarning)
C:\Users\rajaa\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 33.0% of the points cannot be placed; you may want to decrease the size of the markers or use stripp
  warnings.warn(msg, UserWarning)
<AxesSubplot:xlabel='Gender', ylabel='ApplicantIncome'>
```



[ ] #Descriptive Analysis

[ ] data.describe()

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

[ ] #Checking For Null Values

[ ] data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
[ ] #Data preprocessing
```

```
[ ] data.drop(["Loan_ID"], axis=1, inplace=True)
```

```
[ ] data
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 12 columns

```
[ ] #Handling Missing Values
```

```
[ ] def missing_values(df):
        a = num_null_values = df.isnull().sum()
        return a
```

```
[ ] missing_values(data)
```

```
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

```
[ ] #Handling Missing Values
```

```
[ ] # dropping the missing values
    data = data.dropna()
```

```
[ ] data
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
5	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

480 rows × 12 columns

```
[ ] data.isnull().sum()
```

```
Gender              0
Married             0
Dependents          0
Education           0
Self_Employed       0
ApplicantIncome     0
CoapplicantIncome   0
LoanAmount          0
Loan_Amount_Term    0
Credit_History      0
Property_Area       0
Loan_Status         0
```

```
dtype: int64
```

```
[ ] #Encoding
```

```
[ ] from sklearn.preprocessing import OrdinalEncoder

ord_enc = OrdinalEncoder()
data[["Gender","Married','Dependents','Education','Self_Employed','Property_Area','Loan_Status']] = ord_enc.fit_transform(data[["Gender","Married','Dependents','Education','Self_Employ
data.head()
```

```
C:\Users\rajaa\anaconda3\lib\site-packages\pandas\core\frame.py:3678: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[col] = igetitem(value, i)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1.0	1.0	1.0	0.0	0.0	4583	1508.0	128.0	360.0	1.0	0.0	0.0
2	1.0	1.0	0.0	0.0	1.0	3000	0.0	66.0	360.0	1.0	2.0	1.0
3	1.0	1.0	0.0	1.0	0.0	2583	2358.0	120.0	360.0	1.0	2.0	1.0
4	1.0	0.0	0.0	0.0	0.0	6000	0.0	141.0	360.0	1.0	2.0	1.0
5	1.0	1.0	2.0	0.0	1.0	5417	4196.0	267.0	360.0	1.0	2.0	1.0

```
[ ] data[["Gender","Married','Dependents','Education','Self_Employed','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area','Loan_Status']] =data[["Gender","
```

```
C:\Users\rajaa\anaconda3\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

```
[ ] data
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
1	1	1	1	0	0	4583	1508	128	360	1	0	0
2	1	1	0	0	1	3000	0	66	360	1	2	1
3	1	1	0	1	0	2583	2358	120	360	1	2	1
4	1	0	0	0	0	6000	0	141	360	1	2	1
5	1	1	2	0	1	5417	4196	267	360	1	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...
609	0	0	0	0	0	2900	0	71	360	1	0	1
610	1	1	3	0	0	4106	0	40	180	1	0	1
611	1	1	1	0	0	8072	240	253	360	1	2	1
612	1	1	2	0	0	7583	0	187	360	1	2	1
613	0	0	0	0	1	4583	0	133	360	0	1	0

480 rows × 12 columns

```
[ ] #Train Test Split
```

```
X = data.drop("Loan_Status", axis=1)
y = data["Loan_Status"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(384, 11)
(384,)
(96, 11)
(96,)
```

```
[ ] #Naive bayes
```

```
[ ] from sklearn.naive_bayes import GaussianNB

gfc = GaussianNB()
gfc.fit(X_train, y_train)
pred1 = gfc.predict(X_test)
training_data_accuray_nb = accuracy_score(pred1,y_test)
```

```
[ ] training_data_accuray_nb
```

```
0.7708333333333334
```

```
[ ] SVM Classifier
```

```
[ ] from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
```

```
classifier = SVC(kernel='linear')
```

```
[ ] training the support Vector Macine model
    classifier.fit(X_train,y_train)
    X_test_prediction = classifier.predict(X_test)
    test_data_accuray_svm = accuracy_score(X_test_prediction,y_test)
```

```
[ ] test_data_accuray_svm
```

```
[ ] #XGB Classifier
```

```
[ ] from xgboost import XGBClassifier

    xgb = XGBClassifier(learning_rate =0.1,
     n_estimators=1000,
     max_depth=3,
     min_child_weight=1,
     gamma=0,
     subsample=0.8,
     colsample_bytree=0.8,
     objective= 'binary:logistic',
     nthread=4,
     scale_pos_weight=1,
     seed=27)
    xgb.fit(X_train, y_train)
    pred3 = xgb.predict(X_test)
    test_data_accuray_xgb = accuracy_score(y_test,pred3)
```

```
C:\Users\rajaa\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remo
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[14:05:44] WARNING: D:\bld\xgboost-split_1645118015404\work\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was cl
```

```
[ ] test_data_accuray_xgb

    0.7083333333333334
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import RandomizedSearchCV

    def randomized_search(params, runs=20, clf=DecisionTreeClassifier(random_state=2)):
        rand_clf = RandomizedSearchCV(clf, params, n_iter=runs, cv=5, n_jobs=-1, random_state=2)
        rand_clf.fit(X_train, y_train)
        best_model = rand_clf.best_estimator_

        # Extract best score
        best_score = rand_clf.best_score_

        # Print best score
        print("Training score: {:.3f}".format(best_score))

        # Predict test set labels
        y_pred = best_model.predict(X_test)

        # Compute accuracy
        accuracy = accuracy_score(y_test, y_pred)

        # Print accuracy
        print('Test score: {:.3f}'.format(accuracy))

        return best_model
```

```
[ ] ds = DecisionTreeClassifier(max_depth=8, max_features=0.9, max_leaf_nodes=30,
                                min_impurity_decrease=0.05, min_samples_leaf=0.02,
                                min_samples_split=10, min_weight_fraction_leaf=0.005,
                                random_state=2, splitter='random')
    ds.fit(X_train, y_train)
    pred4 =ds.predict(X_test)
    test_data_accuracy_dt = accuracy_score(y_test,pred4)
```

```
[ ] ds
```

```
DecisionTreeClassifier(max_depth=8, max_features=0.9, max_leaf_nodes=30,
                       min_impurity_decrease=0.05, min_samples_leaf=0.02,
                       min_samples_split=10, min_weight_fraction_leaf=0.005,
                       random_state=2, splitter='random')
```

```
[ ] from sklearn.ensemble import RandomForestClassifier

    randomized_search(params={
                            'min_samples_leaf':[1,2,4,6,8,10,20,30],
                             'min_impurity_decrease':[0.0, 0.01, 0.05, 0.10, 0.15, 0.2],
                             'max_features':['auto', 0.8, 0.7, 0.6, 0.5, 0.4],
                             'max_depth':[None,2,4,6,8,10,20],
                            }, clf=RandomForestClassifier(random_state=2))
```

```
Training score: 0.815
Test score: 0.781
```

```
RandomForestClassifier(max_depth=2, max_features=0.5,
                       min_impurity_decrease=0.01, min_samples_leaf=10,
                       random_state=2)
```

```
logisticRegr = LogisticRegression()
logisticRegr.fit(X_train, y_train)
predictions = logisticRegr.predict(X_test)
test_data_accuracy_lr = accuracy_score(y_test,predictions)
```

```
test_data_accuracy_lr
```

```
0.75
```

```
#Knn
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
test_data_accuracy_knn = accuracy_score(y_test,predictions)
```

```
test_data_accuracy_knn
```

```
0.6458333333333334
```

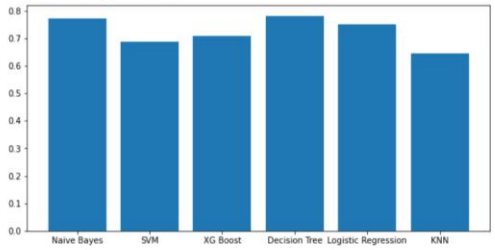
```
data=[training_data_accuray_nb,test_data_accuray_svm,test_data_accuray_xgb,test_data_accuracy_dt,test_data_accuracy_lr,test_data_accuracy_knn]
data = np.array(data)
data2 = data
print(data2)
```

```
[0.77083333 0.6875     0.70833333 0.78125    0.75       0.64583333]
```

```
models = ["Naive Bayes", "SVM", "XG Boost", "Decision Tree", "Logistic Regression", "KNN" ]
```

```
fig = plt.figure(figsize = (10, 5))
plt.bar(models,data2)
```

```
<BarContainer object of 6 artists>
```




```
import joblib
joblib.dump(ds, "model.pkl")
model = joblib.load('model.pkl')
model.predict(X_test)
```

```
array([1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1])
```

Colab paid products - Cancel contracts here